International Journal of Grid and Distributed Computing
Vol. 9, No. 5 (2016), pp.273-282
http://dx.doi.org/10.14257/ijgdc.2016.9.5.23

Load Balancing Through Arranging Task With Completion Time

Palash Samanta[1], Ranjan Kumar Mondal[2]

[1]*Department of Engineering and Technology,*
University of Kalyani, Kalyani, India
[2]*Department of Computer Science & Engineering,*
University of Kalyani, Kalyani, India
[1]*samanta.palsh30@gmail.com*, [2]*ranjan@klyuniv.ac.in*

Abstract

*Nowadays, different types of bandwidth eater are growing rapidly. Cloud computing as an Internet computing has propagate day by day to provide different type of accommodations and resources to web utilizer. Cloud computing employs Internet resources to execute sizably voluminous-scale tasks. Ergo, to cull felicitous node to execute a task is able to enhance the performance of astronomically immense-scale cloud computing environment. There are several different nodes in a cloud computing system. Namely, each node has different capability to execute task; hence, only consider the CPU remaining of the node is not enough when a node is opted to execute a task. Consequently, how to select an efficient node to execute a task is very consequential in a cloud computing.In this paper, we propose a scheduling algorithm, Load **B**alancing through **A**rranging **T**ask with **C**ompletion **T**ime, **LBATCT** which combines minimum completion time and load balancing strategies. For the case study, LBATCT can provide efficient utilization of computing resources and maintain the load balancing in cloud computing environment.*

Keywords*: Cloud Computing, Load Balancing, Distributed System, Scheduling.*

I. Introduction

Cloud Computing became very popular in the last few years. As a component of its accommodations, it provides flexible and facile way to keep and retrieve data and files. Especially for magnifying data sets and files available for the spreading number of users around the world. Handling such immensely colossal data sets require several techniques to optimize and streamline operations and provide copacetic levels of performance for the users.

Ergo, it is paramount to research some areas in the Cloud to amend the storage utilization and the download performance for the users. One paramount issue associated with this field is dynamic load balancing or task scheduling. Load balancing algorithms were investigated heavily in different environments; however, with Cloud environments, some real challenges are present and must be addressed. In Cloud Computing the main concerns involve efficiently assigning tasks to the Cloud nodes such that the effort and request processing is done as efficiently as possible [1], while being able to abide the sundry affecting constraints such as heterogeneity and high communication delays. Load balancing algorithms are classified as static and dynamic algorithms. Static algorithms are suitable for homogeneous and stable environments and can produce very good results in these environments. Dynamic algorithms are more flexible and take into consideration different types of attributes in the system both prior to and during run-time. These type of algorithms can adapt to changes and provide better results in heterogeneous and dynamic environments. However, as the distribution attributes become more complex and dynamic.

ISSN: 2005-4262 IJGDC

II. Challenges in Cloud Computing Load Balancing

Afore we could review the current load balancing approaches for Cloud Computing, we require to identify the cognate issues and challenges involved and that could affect how the algorithm would perform. Here we discuss the challenges to be addressed when endeavoring to propose an optimal solution to the issue of load balancing in Cloud Computing. These challenges are summarized in the following points.

A. Spatial Distribution of the Cloud Nodes

Some algorithms are designed to be efficient for that nodes where communication delays are negligible. However, it is an issue to design a load balancing algorithm that can work opportunely for spatially distributed nodes. This is because other factors must be taken into account such as the celerity of the network links among the nodes, the distance between the client and the task processing nodes, and the distances between the nodes involved in providing the accommodation. There is a desideratum to develop a way to control load balancing mechanism among all the spatial distributed nodes while being able to efficaciously abide high delays [3].

B. Storage/ Replication

A full replication algorithm does not take efficient storage utilization into account. This is because the same data will be stored in all replication nodes. Full replication algorithms impose higher costs since more storage is needed. However, partial replication algorithms could save parts of the data sets in each node (with a certain level of overlap) based on each node's capabilities such as processing power and capacity [4]. This could lead to better utilization, yet it increases the complexity of the load balancing algorithms as they attempt to take into account the availability of the data set's parts across the different Cloud nodes.

C. Algorithm Complexity

Load balancing algorithms are preferred to be less complex in terms of implementation and operations. The higher implementation complexity would lead to a more complex process which could cause some negative performance issues. Furthermore, when the algorithms require more information and higher communication for monitoring and control, delays would cause more problems and the efficiency will drop. Therefore, load balancing algorithms must be designed in the simplest possible forms [5].

D. Point of Failure

Controlling the load balancing and accumulating data about the different nodes must be designed in a way that evades having a single point of failure in the algorithm. Some algorithms (centralized algorithms) can provide efficient and efficacious mechanisms for solving the load balancing in a certain pattern. However, they have the issue of one controller for the whole system. In such cases, if the controller fails, then the whole system would fail. Any Load balancing algorithm must be designed in order to surmount this challenge [6]. Distributed load balancing algorithms seem to provide a better approach, yet they are much more involute and require more coordination and control to function correctly.

III. Load Balancing Algorithms Review

In this section we discuss the most kenned contributions in the literature for load balancing in Cloud Computing. We relegate the load balancing algorithms into two types: static algorithms and dynamic algorithms. We first discuss the static load-balancing

algorithms that have been developed for Cloud Computing. Then, we will discuss the dynamic load-balancing algorithms.

A. Static Load Balancing Algorithms

Static Load balancing algorithms assign the tasks to the nodes predicated only on the faculty of the node to process incipient requests. The process is predicated solely on prior erudition of the nodes' properties and capabilities. These would include the node's processing puissance, recollection and storage capacity, and most recent kenned communication performance. Albeit they may include erudition of the communication prior performance, static algorithms generally do not consider dynamic changes of these attributes at run-time. In integration, these algorithms cannot habituate to load changes during run-time. Radojevic suggested an algorithm called CLBDM [7] (Central Load Balancing Decision Model). CLBDM is an amendment of the Round Robin Algorithm which is predicated on session switching at the application layer. Round Robin [8] is a very famous load balancing algorithm. However, it sends the requests to the node with the least number of connections. The amelioration done in CLBDM is that the connection time between the client and the node in the cloud is calculated, and if that connection time exceeds a threshold then there is an issue. If an issue is found, the connection will be terminated and the task will be forwarded to another node utilizing the conventional Round Robin rules. CLBDM acts as an automated administrator. The conception was inspired by the human administrator perspective.

The proposed algorithm by Kumar [9] is an amelioration version of the algorithm presented in [10]. Both algorithms are utilizing the ants' comportment to accumulate information about the cloud nodes to assign the task to a concrete node. However, the algorithm in [10] has the ant's synchronization issue and the author in [9] is endeavoring to solve this by integrating the feature 'suicide' to the ants. Both algorithms work in the following way, once a request is initiated the ants and pheromone are initiated and the ants start their forward path from the 'head' node. A forward kineticism denotes that the ant is peregrinating from one overloaded node probing for the next node to check if it is overloaded or not. Moreover, if the ant finds an under loaded node, it will perpetuate its forward path to check the next node.

If the next node is an overloaded node, the ant will utilize the rearward kineticism to get to the antecedent node. The additament in the algorithm proposed in [9] that is ant colony algorithm.

The algorithm proposed in [11] is a MapReduce algorithm [12]. MapReduce is a model which has two main tasks: It Maps tasks and Reduces tasks results.

Junjie proposed a load balancing algorithm [13] for the Cloud utilizing VM to physical machine mapping. The architecture of the algorithm contains a central scheduling controller and a resource monitor. The scheduling controller does all the work for calculating which resource is able to take the task and then assigning the task to that concrete resource.

B. Dynamic Load Balancing Algorithms

Dynamic load balancing algorithms take into account the different attributes of the nodes' capabilities and network bandwidth. Most of these algorithms rely on a cumulation of erudition predicated on prior amassed information about the nodes in the Cloud and run-time properties amassed as the culled nodes process the task's components. These algorithms assign the tasks and may dynamically reassign them to the nodes predicated on the attributes accumulated and calculated. Such algorithms require constant monitoring of the nodes and task progress and are customarily harder to implement. However, they are more precise and could result in more efficient load balancing.

In [14], the goal is to find an algorithm to minimize the data duplication and redundancy. The algorithm proposed is called INS (Index Name Server) and it integrates duplication and access point cull optimization. There are many parameters involved in the process of calculating the optimum cull point. Some of these parameters are the Hash code of the block of data to be downloaded, the position of the server that has the target block of data, the transition quality which is calculated predicated on the node performance and a weight judgment chart, the maximum bandwidth of downloading from the target server and the path parameter.

Another calculation is utilized to ascertain whether the connection can handle supplemental nodes or not (diligent level). They relegated the diligent levels into three main categories B(a), B(b) and B(c).

B(a) betokens that the connection is very diligent and cannot handle any adscititious connections. B(b) denotes the connection is not diligent and supplemental connections can be integrated. However, B(c) betokens that the connection is inhibited and further study needs to be done to ken more about the connection. B(b) is withal relegated into three categories: B(b1) which designates that INS must analyze and establish a backup, B(b2) which betokens the INS must send the requests to the backup nodes and B(b3) which is the highest caliber of efficiency required and it signifies that INS must reanalyze and establish incipient backups.

Ren [15] presented a dynamic load balancing algorithm for cloud computing predicated on a subsisting algorithm called WLC [16] (weighted least connection). The WLC algorithm assigns tasks to the node predicated on the number of connections that subsist for that node. This is done predicated on a comparison of the SUM of connections of each node in the Cloud and then the task is assigned to the node with least number of connections.

However, WLC does not take into consideration the capabilities of each node such as processing celerity, storage capacity and bandwidth. The proposed algorithm is called ESWLC (Exponential Smooth Forecast predicated on Weighted Least Connection). ESWLC ameliorates WLC by taking into account the time series and tribulations. That is ESWLC builds the conclusion of assigning a certain task to a node after having a number of tasks assigned to that node and getting acquainted with the node capabilities. ESWLC builds the decision predicated on the experience of the node's CPU potency, recollection, number of connections and the amount of disk space currently being utilized.

ESWLC then soothsays which node is to be culled predicated on exponential smoothing.

The algorithm proposed in [17] is a dual direction downloading algorithm from FTP servers (DDFTP). The algorithm presented can be additionally implemented for Cloud Computing load balancing. DDFTP works by splitting a file of size m into m/2 partitions. Then, each server node commences processing the task assigned for it predicated on a certain pattern.

For example, one server will commence from block 0 and keeps downloading incrementally while another server commences from block m and keeps downloading in a decrement order. As a result, both servers will work independently, but will culminate up downloading the whole file to the client in the best possible time given the performance and properties of both servers.

Thus, when the two servers download two consecutive blocks, the task is considered as culminated and other tasks can be assigned to the servers. The algorithm reduces the network communication needed between the client and nodes and ergo reduces the network overhead. Moreover, attributes such as network load, node load, network speed are automatically taken into consideration, while no run-time monitoring of the nodes is required.

The paper in [18] proposes an algorithm called Load Balancing Min-Min (LBMM). LBMM has a three level load balancing framework. It utilizes the Opportunistic Load

Balancing algorithm (OLB) [19]. OLB is a static load balancing algorithm that has the goal of keeping each node in the cloud diligent. However, OLB does not consider the execution time of the node. This might cause the tasks to be processed in a more gradual manner and will cause some bottlenecks since requests might be pending waiting for nodes to be free.

LBMM ameliorates OLB by integrating a three layered architecture to the algorithm. The first level of the LBMM architecture is the request manager which is responsible for receiving the task and assigning it to one accommodation manager in the second level of LBMM. When the accommodation manager receives the request, it divides it into subtasks to expedite processing that request. An accommodation manager would additionally assign the subtask to an accommodation node which is responsible for executing the task. The accommodation manager assigns the tasks to the accommodation node predicated on different attributes such as the remaining CPU space (node availability), remaining memory and the transmission rate.

3. The Proposed Method

There are several different nodes in a cloud computing system. Namely, each node has different capability to execute task; hence, only consider the CPU remaining of the node is not adequate when a node is opted to execute a task. Consequently, how to cull a particular node to execute that task is very paramount issue in a cloud computing area.

Due to the task that has different characteristic for utilizer to pay execution. Hence it is need some of the resources of categorical, for instance, when implement organism sequence assembly, it is probable have to sizably voluminous requisite toward recollection remaining. And in order to reach the best efficient in the execution each tasks, so we will aimed by tasks property to adopt a different condition decision variable in which it is according to resource of task requisite to set decision variable.

Problem Definition:

Due to the problem of the resource allocation with minimum completion time of a task with reasonably low cost. In this paper we are going to discussed some proposed algorithms have been developed under some assumptions.

Method and Case Study:

The progress of Load Balancing through arranging task with completion time is presented as following:

Step 1: First we have to calculate the average completion time of entire table for all tasks.
Step 2: We have to arrange the completion time of every task in ascending or descending order (here we are doing in ascending order) in another table.
Step 3: Now we can say that the entire task cannot be completed within the maximum time of the minimum column of the new table.
Step 4: Now we have to do one more thing that is we assign the task to the node in descending order those are taking less time than the average time. And assign the task in ascending order those are taking more time than the average time.
Step 5: We assign the task through first column to the last column. Assign first, maximum task those are less than average time (if) else assign minimum task those are taking more time than the average time. If already assign that node then cross mark it.
Step 6: If clash the time then pick the time which is taking more time to another machine by looking the next column of the new table else assign and go to the step 5.
Step 7: Upto this we have no problem, we can again reduce the time if possible.
As an example we will not assign the maximum task less than average time immediately. We will keep it up to less than maximum time and find possible combination as shown in

the illustrated example.

Method and Case Study:

Now we are showing the progress of Load Balancing through arranging task with completion time that is presented as following:

Step 1: First we have to calculate the average completion time of entire table for all tasks. As shown in table 2 which is the average of the table 1.

Step 2: Then we have to do one more important work as shown in table 3.
We have to arrange the completion time of every task in ascending or descending order (here we are doing in ascending order).

Step 3: Now we have to assign each node with a task which will take minimum total completion time. We can see that the average completion time is 25.94, so our requirement is the result will be near about the average time or less than it.

Step 4: First we will assign task to the node from first column from table 3. Only for the first column we will assign task with higher completion time to lower completion time (as for example for first column we will assign task as 19, 18 14, 12 in this order and for other columns we will assign task with lower completion time to higher completion time).

Step 5: Now we can see that for the first column we can assign 19 of T_4 in C_{11} machine (from table 1). We will not assign it immediately because we may get a better result. So we will mark it. Now assign other task which is not in the same node (First assign 18 of T_3 of C_{12} machine). Find other task from first column of table 3. Now we can see that the other tasks cannot assign from column 1 (table 3) because all are in the same node (table 1). So we ignore all.

Step 6: for second column to the last column(from table 3) assign task from lower completion time to higher completion time (which task are taking more time than average time otherwise task are assign from higher completion time to lower completion time(whose completion time are less than average time)) **.**

Here we can see that (from table 3) only 26 is the time greater than average time all of others are less than average time. So we first see the time for 20 seconds, again we will not assign immediately because we may get better result. So mark it. Then go to 18 seconds T_1 of C_{11} machine (from table 1) we are getting better result so assign it and cut the 19 seconds. Repeat it for 14 seconds T_2 of C_{11} machine, which is already assign job so ignore it. And for 26 seconds of T_3 of C_{11} machine, which is already assign so ignore it.

Step 7: Repeat step 6 until all tasks have completed totally.
Here we can see that (for third column of table 3) 18 and 24 are less than average time and others are greater than average time. So first in case of 24 seconds it again mark for better result. Now for 18 seconds T_2 of C_{14} machine, assign it.
Now for others 26 and 42 seconds all task are assign already. So ignore it.

Step 8: Repeat step 6 until all tasks have completed totally.
Here we can see that (for fourth column of table 3) only 24 seconds are less than average time but T_2 is already assign so ignore it. In case of others we will go through lower completion time to higher completion time (all are taking more than average time). First in case of 36 seconds which is much more than the average time so ignore it. All others will automatic ignore, because all are assign already.

Step 9: At the end we are still not assign any machine for T_4 task.
We can look the marking result (from table 3) that only the 24 sec of C_{13} will give the better result. So assign it and get the total maximum completion time is 24 seconds. The completion time for MM, Max Min, LBMM, LB3M are 44, 37, 33, 26 respectively. But

in new approach it is taking to complete is 24 seconds.

Table 1. Completion Time (Second) of each Node at Different Tasks

Task \ Node	C_{11}	C_{12}	C_{13}	C_{14}
T_1	18	14	38	26
T_2	14	12	24	18
T_3	26	18	66	42
T_4	19	20	24	36

Table 2. Average Value of all Tasks

Average	25.9375

Table 3. Completion Time (Second) of each Node at Different Tasks

Task \ Node	C_{11}	C_{12}	C_{13}	C_{14}
T_1	14	**18**	26	38
T_2	12	14	**18**	24
T_3	**18**	26	42	66
T_4	19	20	**24**	36

Table 4. The Comparison of Completion Time of each Task at Different Node for Case Study

Task \ Node	C_{11}	C_{12}	C_{13}	C_{14}
T_1	**18**	14	38	26
T_2	14	12	24	**18**
T_3	26	**18**	66	42
T_4	19	20	**24**	36

6. Comparison

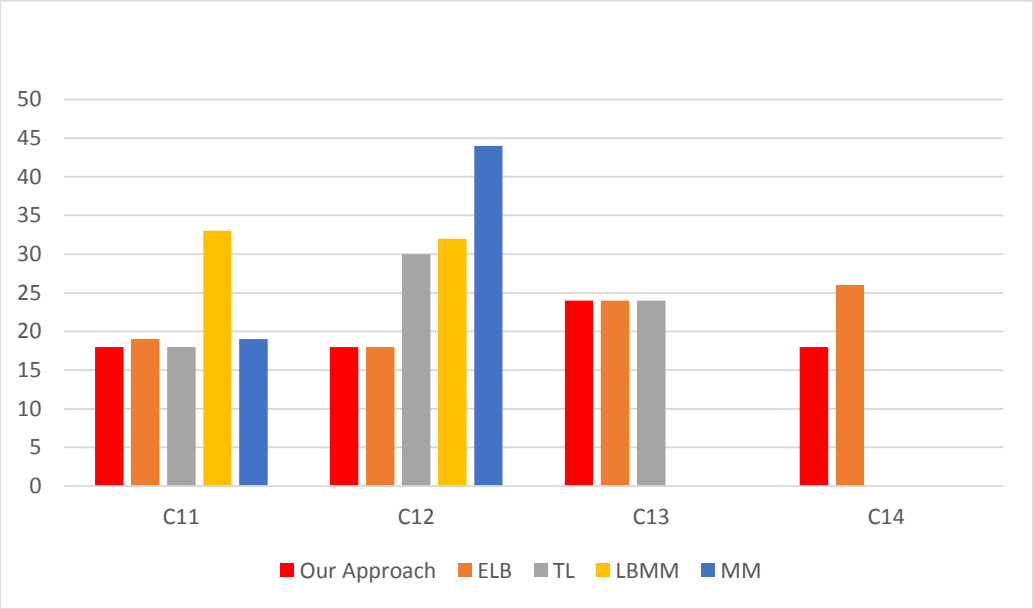


Figure 1.The Comparison of Completion Time of each Task at Different Node for Case Study

7. Conclusion

In this paper, we proposed an efficient scheduling algorithm, LBATCT, for the cloud computing network to assign tasks to computing nodes according to their resource capability. However, the load balancing of cloud computing network is utilized, all calculating result could be integrated first by the second level node before sending back to the management. Thus, the goal of loading balance and better resources manipulation could be achieved.

Acknowledgement

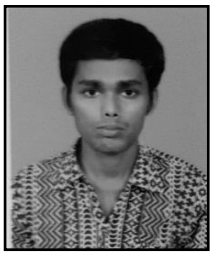
We would like to express our gratitude to Debabrata Sarddar, Assistant Professor, Department of Computer Science and Engineering of University of Kalyani. Without his assistance and guidance, we would not have been able to make use of the university's infrastructure and laboratory facilities for conducting our research.

References

[1] Randles, M., D. Lamb and A. Taleb-Bendiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Perth, Australia, April 2010.
[2] Rimal, B. Prasad, E. Choi and I. Lumb, "A taxonomy and survey of cloud computing systems." In proc. 5th International Joint Conference on INC, IMS and IDC, IEEE, 2009.
[3] Buyya R., R. Ranjan and RN. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in proc. 10th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP), Busan, South Korea, 2010.
[4] Foster, I., Y. Zhao, I. Raicu and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," in proc. Grid Computing Environments Workshop, pp: 99-106, 2008.
[5] Grosu, D., A.T. Chronopoulos and M. Leung, "Cooperative load balancing in distributed systems," in Concurrency and Computation: Practice and Experience, Vol. 20, No. 16, pp: 1953-1976, 2008.
[6] Ranjan, R., L. Zhao, X. Wu, A. Liu, A. Quiroz and M. Parashar, "Peerto-peer cloud provisioning: Service discovery and load-balancing," in Cloud Computing - Principles, Systems and Applications, pp: 195-217, 2010.
[7] Radojevic, B. and M. Zagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments." In proc.34th International Convention on MIPRO, IEEE, 2011.
[8] Sotomayor, B., RS. Montero, IM. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," in IEEE Internet Computing, Vol. 13, No. 5, pp: 14-22, 2009.
[9] Nishant, K. P. Sharma, V. Krishna, C. Gupta, KP. Singh, N. Nitin and R. Rastogi, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization." In proc. 14th International Conference on Computer Modelling and Simulation (UKSim), IEEE, pp: 3-8, March 2012.
[10] Zhang, Z. and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation." In proc. 2nd International Conference on. Industrial Mechatronics and Automation (ICIMA), IEEE, Vol. 2, pp:240-243, May 2010.
[11] Kolb, L., A. Thor, and E. Rahm, E, "Load Balancing for MapReducebased Entity Resolution," in proc. 28th International Conference on Data Engineering (ICDE), IEEE, pp: 618-629, 2012.
[12] Gunarathne, T., T-L. Wu, J. Qiu and G. Fox, "MapReduce in the Clouds for Science," in proc. 2nd International Conference on Cloud Computing Technology and Science (CloudCom), IEEE, pp:565-572, November/December 2010.
[13] Ni, J., Y. Huang, Z. Luan, J. Zhang and D. Qian, "Virtual machine mapping policy based on load balancing in private cloud environment," in proc. International Conference on Cloud and Service Computing (CSC), IEEE, pp: 292-295, December 2011.
[14] T-Y., W-T. Lee, Y-S. Lin, Y-S. Lin, H-L. Chan and J-S. Huang, "Dynamic load balancing mechanism based on cloud storage" in proc. Computing, Communications and Applications Conference (ComComAp), IEEE, pp:102-106, January 2012.
[15] Ren, X., R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast" in proc. International Conference on. Cloud Computing and Intelligent Systems (CCIS), IEEE, pp: 220-224, September 2011.
[16] Lee, R. and B. Jeng, "Load-balancing tactics in cloud," in proc. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, pp:447-454, October 2011.
[17] Al-Jaroodi, J. and N. Mohamed. "DDFTP: Dual-Direction FTP," in proc. 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), IEEE, pp:504-503, May 2011.

[18] Wang, S-C., K-Q. Yan, W-P. Liao and S-S. Wang, "Towards a load balancing in a three-level cloud computing network," in proc. 3rd International Conference on. Computer Science and Information Technology (ICCSIT), IEEE, Vol. 1, pp:108-113, July 2010.
[19] Sang, A., X. Wang, M. Madihian and RD. Gitlin, "Coordinated load balancing, handoff/cell-site selection, and scheduling in multi-cell packet data systems," in Wireless Networks, Vol. 14, No. 1, pp: 103-120, January 2008.
[20] Mohamed, N. and J. Al-Jaroodi, "Delay-tolerant dynamic load balancing," in proc. 13th International Conference on High Performance Computing and Communications (HPCC), pp:237-245, September 2011.
[21] Ranjan Kumar Mondal, Payel Ray, Debabrata Sarddar "Load Balancing", *International Journal of Research in Computer Applications & Information Technology*, Volume 4, Issue 1, January-February, 2016, pp. 01- 21, ISSN Online: 2347- 5099, Print: 2348- 0009, DOA : 03012016.
[22] Ranjan Kumar Mondal, Enakshmi Nandi, and Debabrata Sarddar. "Load Balancing Scheduling with Shortest Load First." *International Journal of Grid and Distributed Computing* 8.4 (2015): 171-178.
[23] Ranjan Kumar Mondal, Payel Ray, Debabrata Sarddar "Load Balancing with Task Division and Addition". *International Journal of Scientific Research Engineering & Technology* (IJSRET), ISSN 2278 – 0882, Volume 5, Issue 1, January 2016: 15-19.
[24] Ranjan Kumar Mondal, Debabrata Sarddar "Load Balancing with Task Subtraction of Same Nodes". International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online) Vol. 3, Issue 4, pp: (162-166), Month: October - December 2015.
[25] Ranjan Kumar Mondal, Payel Ray, Enakshmi Nandi, and Debabrata Sarddar. "Load Balancing with Minimum Task." *IJournals: International Journal of Software & Hardware Research in Engineering*, ISSN-2347-4890, Volume 4 Issue 1 January, 2016: 37-41.
[26] Ranjan Kumar Mondal, Payel Ray, Enakshmi Nandi, Debabrata Sarddar, "Load Balancing Algorithm with Total Task of Different Nodes." *IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS)*, ISSN: 2249-9555, Vol.6, No1, Jan-Feb 2016.
[27] Ranjan Kumar Mondal, Enakshmi Nandi, Payel Ray, Debabrata Sarddar, "Load balancing with Modify Approach", *International Journal of Computer Techniques*. ISSN: 2394-2231, Volume 3 Issue 1, Jan-Feb 2015, pp. 1-5.
[28] Ranjan Kumar Mondal, Payel Ray, Enakshmi Nandi, Debabrata Sarddar, "Load Balancing with Tasks Subtraction" International Journal of Science, *Engineering and Technology Research (IJSETR)*, ISSN:2278 –7798, Volume 5, Issue 1, January 2016, pp-336-344.

Authors



Palash Samanta is currently pursuing B.Tech in Information Technology from Department of Engineering and Technology in University of Kalyani, Kalyani, Nadia, West Bengal, India. His research interests include Cloud Computing, Wireless and Mobile Communication Systems.

Ranjan Kumar Mondal received his M.Tech in Computer Science and Engineering from University of Kalyani, Kalyani, Nadia; and B.Tech in Computer Science and Engineering from Government College of Engineering and Textile Technology, Berhampore, Murshidabad, West Bengal under West Bengal University of Technology, West Bengal, India. At present, he is a Ph.D research scholar in Computer Science and Engineering from University of Kalyani. His research interests include Cloud Computing, Wireless and Mobile Communication Systems.

